Contacts Manager

HCI 2019-2020 Programming Assignment

Prof. Andrew D. Bagdanov

November 17, 2019

1 Overview

We are **flooded** with torrents of digital communication in our daily lives. Email, telephone calls, instant messaging, Skype calls, Tweets, and a myriad of other communication sources contribute to a sense of **overload**. Not only must we manage this **communication**, we must also keep track of the various **contacts** with whom we communicate. A **Contacts Manager** does just that for us: it maintains a list of known contacts with all of their important contact information (names, numbers, addresses, notes, etc).

In this programming assignment you will implement a simple contact manager as a complete and polished Graphical User Interface. This GUI application will support the basic operations needed for a contact manager.

2 Assignment

For this assignment you must implement a graphical Contacts Manager in your programming language of choice. You may implement this in **any programming language** and using **any GUI framework** you desire. Almost anything goes, however the features your GUI must implement to receive full credit your GUI must implement the following features:

- **Visualization of all contacts**: your GUI must support visualization of the **entire** contacts database. Contact records should support (as a minimum): first and last names, email address, phone number, free-text notes. The visualization of contacts must allow the user to **order** the list of contacts by field.
- **Single-contact visualization**: your GUI must allow the user to view a single contact. This visualization should give the option to **delete** the contact.
- **Insertion of new contact**: your GUI must allow the user to **create** a new contact. The interface must allow the user to insert data for all (or some) fields and save the new contact.
- **Contact persistence**: your GUI must **save** the contact list across multiple invocations of the application.

Your GUI can also support the following features (which will be considered as **extra credit** in the final evaluation:

- **Contact tagging and tag search**: allow the user to associate **tags** to contacts and allow them to filter contact visualization by **tag**.
- **Contact editing**: when the user **views** a contact, allow them to **edit** it as well.
- **Full-text search**: allow the user to enter **search terms** which are then used to filter contact visualization to contacts with those terms in one or more fields.

- **Dynamic persistence**: allow users to **asynchronously** edit the contacts database from **multiple instances** of your GUI. Active GUIs should dynamically update whenever changes are made.

3 Evaluation

You **must** submit the source code for your complete implementation by the **deadline for registering for the exam**. Late submissions will **NOT** be accepted.

A note on **cheating**. I realize that there are **many** implementations of the Contact Managers out there. That is great, and you should look at them and learn from them. However, the code you submit **MUST BE YOUR OWN WORK**. Learn from others, but **write and submit your own implementation**.

Your implementation will be evaluated based on how you apply the programming models and constructs learned during the course. More specifically:

- **Functionality** (25%): your code must **work**. In your submission, include a minimal README that explains how to run your program (including any dependencies).
- **Code hygiene** (25%): keep things **clear**. This means writing concise, clear, and **reasonably documented** code. You should carefully identify **model**, **view**, and (maybe) **controller** components of your implementation.
- **Completeness** (50%): did you implement all of the **required** features? See the list above, but your GUI must implement all of the required features to earn full credit.
- **Extras** (max 10% bonus points): above and beyond. If you implement **extra features**, make sure you highlight them in the documentation for your submitted project.

4 Hints

Here I will collect some useful hints and advice for anyone choosing this programming assignment (this list might be updated, check back):

- The focus of this assignment is on **complete** and **correct** implementation using **best practices** (i.e. MVC). Your interface does not have to be **pretty**. Make it all **work**, make it work **correctly**, and **then** make it pretty if you have time.
- When it comes to serialization formats, **keep it simple**. For a simple prototype like this, usually a basic, text-based format (e.g. **CSV**) is more than adequate. If you use Python, the **pickle** (or cPickle) library can be extremely useful for serialization.
- If you want to use a **DBMS** as a serialization backend, still keep it super simple and use sqlite (but consider also using a lightweight ORM).

5 Useful Links

The **Contacts Manager** is a classic programming assignment for introductory GUI programming courses. There are lots of tutorials out there. Do some Googling. But anyway:

- The Qt QtContacts Module is a good example of how contacts are managed in **the real world**. Take a look at it for inspiration.
- Also, this simple Qt phonebook tutorial is a great source of inspiration for the widgets, dialogs, and interfaces you will have to implement here.